Mark Gameng

Homework 3

Question 3.1 (30 Points) Build a conceptional model for a Movie Ticketing System. The solution should be presented as an ER-diagram. Base your design on the following requirements.

• The database should record information about Customer, Movies, Employee, Feedback, MovieTheater, Booking and Payments.

• A movie is identified by its name and has an ageRating, and genre.

• A Customer has a name which consists of firstName, middleName and lastName. Customers are identified by a unique customerID. A Customer has an Address, phoneNo and email address – Customers can place any number of Bookings (including none). Customer may provide Feedback for each Booking they have placed (optional). For every Booking, a Booking must make a Payment. Customer is associated with at least one address and multiple customer can live in the same place.

• A MovieTheater is identified by it is name and address ID and has capacity and reviewRate

• A Booking is uniquely identified by the bookingId. A Booking is created by a Customer. For each booking we store a total Number of ticket, and a price each ticket. – A Booking is associated with one or more ticket. For each ticket, we must record how many of each ticket type, for example (1 Adult, 1 senior and junior). A feedback includes 3 question a rating for each question (1-5)

• Employee is identified by an employee number. An employee works in one or more MovieTheater. An employee has a name and address and birth year. In each MovieTheater you have a permanent employee taking a yearly salary and temporary employee working on hourly rates.

• An Address consists of a unique addrID, street and has streetNumber, city, state and zipcode. The attributes city and state can be derived from the attribute zipcode. – There may be some Addresses which are not be associated with any Customer or MovieTheater.

• A Payment is identified by the Booking for which the payment was made. It consists of the amountPaid and paymentMethod (Credit Card, E-Check, etc.)

A close up of text on a white background

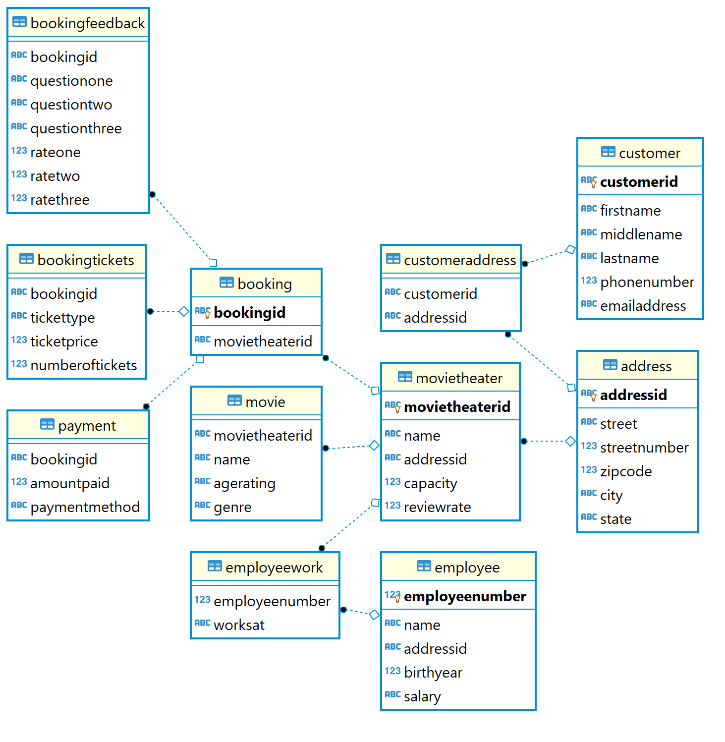
Description automatically generated

Problem 3. 2 (20 points)

Take the ER-model create in problem1 and translate it into a relational schema using the rules presented in class. Present the relational schema as an SQL script. Present the results of the following intermediate steps in this order (write which rule you are using in your translation).

Translate strong entities (including the ISA relation) + unnest composite attributes

1. Translate weak entities
2. Translated multi-valued attributes
3. Translate relationships

from dbeaver

--rule 1,3,7

**create** **table** Address(

addressID **varchar**(255), street **varchar**(255) **not** **null**,

streetNumber **int** **not** **null**, zipcode **int** **not** **null**,

city **varchar**(255) **not** **null**, state **varchar**(255) **not** **null**,

**primary** **key** (addressID)

);

--rule 1,3,4

**create** **table** movieTheater(

movieTheaterID **varchar**(255), **name** **varchar**(255) **not** **null**,

addressID **varchar**(255), capacity **int** **check** (capacity > 0),

reviewRate **int**, **primary** **key** (movieTheaterID),

**foreign** **key** (addressID) **references** Address(addressID)

);

--rule 1,4

**create** **table** Employee(

employeeNumber **int**, **name** **varchar**(255) **not** **null**,

addressID **varchar**(255), birthYear **int** **check**(birthYear > 0),

salary **varchar**(255) **check**(salary **like** '$%' **and** (salary **like** '%/hr' **or** salary **like** '%/year')),

**primary** **key** (employeeNumber)

);

--another table for employee and where they work because employee can work at multiple theaters

--rule 2,3,7

**create** **table** EmployeeWork(

employeeNumber **int**, worksAt **varchar**(255),

**foreign** **key** (employeeNumber) **references** Employee(employeeNumber),

**foreign** **key** (worksAt) **references** movieTheater(movieTheaterID)

);

--rule 1,3,4

**create** **table** Customer(

customerID **varchar**(255), firstName **varchar**(255) **not** **null**,

middleName **varchar**(255) **not** **null**, lastName **varchar**(255) **not** **null**,

phoneNumber **int** **not** **null**,

emailAddress **varchar**(255) **check** (emailAddress **like** '%@%'),

**primary** **key** (customerID)

);

--another table for customer address, because customer can have multiple address

--rule 2,3,7

**create** **table** CustomerAddress(

customerID **varchar**(255), addressID **varchar**(255),

**foreign** **key** (customerID) **references** Customer(customerID),

**foreign** **key** (addressID) **references** Address(addressID)

);

--rule 1,4

**create** **table** Booking(

bookingID **varchar**(255), movieTheaterID **varchar**(255),

**primary** **key** (bookingID),

**foreign** **key** (movieTheaterID) **references** movieTheater(movieTheaterID)

);

--booking has tickets, each ticket has a type and price and amount of tickets

--rule 2,3,7

**create** **table** BookingTickets(

bookingID **varchar**(255), ticketType **varchar**(255) **not** **null**,

ticketPrice **int** **check**(ticketPrice > 0) **not** **null**,

numberOfTickets **int** **check**(numberOfTickets > 0) **not** **null**,

**foreign** **key** (bookingID) **references** Booking(bookingID)

);

--booking also has feedback, which consists of questions and the rate for each

--rule 2,3,7

**create** **table** BookingFeedback(

bookingID **varchar**(255), questionOne **varchar**(255) **not** **null**,

questionTwo **varchar**(255) **not** **null**, questionThree **varchar**(255) **not** **null**,

rateOne **int** **check**(rateOne > 0 **and** rateOne < 6),

rateTwo **int** **check**(rateTwo > 0 **and** rateTwo < 6),

rateThree **int** **check**(rateThree > 0 **and** rateThree < 6),

**foreign** **key** (bookingID) **references** Booking(bookingID)

);

--rule 2, 3

**create** **table** Payment(

bookingID **varchar**(255), amountPaid **int** **check**(amountPaid > 0) **not** **null**,

paymentMethod **varchar**(255) **not** **null**,

**foreign** **key** (bookingID) **references** Booking(bookingID)

);

--rule 2,4

**create** **table** Movie(

movieTheaterID **varchar**(255), **name** **varchar**(255) **not** **null**,

ageRating **varchar**(255) **not** **null**, genre **varchar**(255) **not** **null**,

**foreign** **key** (movieTheaterID) **references** MovieTheater(movieTheaterID)

);